REMARKS

This Amendment is in response to the Office Action of February 13, 2006, in which claims 1-21 were rejected. With this Amendment, claim 18 has been canceled and rewritten as new claim 22. Claims 1-17 and 19-22 are presented for reconsideration and allowance.

In the Office Action, claim 18 was rejected under 35 U.S.C. § 112, second paragraph as being indefinite. The basis of the rejection appears to be the fact that claim 18 depended from a higher numbered claim (claim 21). To overcome the rejection, claim 18 has been canceled and replaced by new claim 21. Claim 22 is identical in content to canceled claim 18, except that it follows claim 21 in numerical order.

Claims 1-5 and 8-21 were rejected under 35 U.S.C. § 102(e) as being anticipated by Rice III (U.S. Pub. 2002/0174010), and dependent claims 6 and 7 were rejected under 35 U.S.C. § 103(a) over Rice in view of Lloyd (U.S. Patent No. 6,779,178). The present invention, as defined in claims 1-17 and 19-22 is neither taught nor suggested by Rice, by Lloyd or by any combination of those two references.

The present invention deploys applications over a distributed network to an internet-enabled device in a manner which is fundamentally different from Rice. In the present invention, text files are downloaded from a server to the device. These text files (or application logic files) contain embedded application logic. The device includes an application assembler (or program assembler) that runs on the device and which downloads the text files from the server. The assembler retrieves the embedded program logic from the downloaded text files, assembles the retrieved program logic into a functioning application, and runs the functioning application on the device. By the way in which the application is assembled at the device, the application program can run on the device regardless of whether the device remains connected to the server.

In contrast, Rice does not download text files, extract embedded program logic and assemble that logic into an application program to run. Rather, Rice downloads executable program code

from the server. That executable code is for the same programs that are running on the server, which allows the device in Rice to run those same programs when the device is not connected to the remote server.

Because Rice does not teach creating applications at the device by downloading text files containing embedded application logic, retrieving the embedded program logic, assembling the retrieved program logic into an application, and then running that application on the device, Rice does not anticipate independent claims 1, 8, or 16, or any of the dependent claims.

On page 10 (section 10, 2nd paragraph), the Office Action incorrectly summarizes Applicant's arguments by saying: "Applicant essentially argues that Rice III does not disclose distributing applications to client (or Internet-enabled) devices". This is not the essence of Applicant's argument or Applicant's claims.

The statement in the Office Action incorrectly over-generalizes Applicant's claims and does not address the many special aspects of that invention that, taken as a whole, make it new, unique and non-obvious when compared to the prior art. The claimed invention is a specific way of distributing applications to client devices, but it is not the only way. Applicant is claiming one specific new way to do this, that is not taught by Rice. Rice did not invent "distributing applications to Internet-enabled devices." ActiveX objects, Java applets and other web browser plugins have done that for many years before Rice. Rice describes a specific way of distributing applications by downloading executable code which is not what Applicant teaches or suggests.

First, Rice does not teach the **downloading of a text file containing program logic** that is then **assembled by an assembler (i.e., an Application Virtual Machine or AVM) into an application** that is **run by an AVM on the local device**. Rice refers to **downloading of executable code**, preferably executable code for a **terminal emulator** that allows the user to interact with **applications running on a remote server**.

Second, Rice discusses the continued downloading of **executable code** comprising the same "fat" programs that are running on the server. That allows the user to run those same programs on the local device when not connected to the remote server. Downloading executable code is well-known prior art.

Applicant's invention does not utilize this mode of operation in any manner. It does not download executable program code for the application. It downloads text files that contain abstract program logic definitions which are used by the AVM to assemble a working application on the local device. Applicant does not require a terminal emulator that is used in conjunction with application programs running on a server, as Rice does.

In Applicant's invention, the only point that involves downloading executable code is the AVM itself, which is not an application. Rather, the AVM is an application assembler/runner, comprising a program that retrieves text files, extracts abstract program logic definitions from the text files, assembles applications from the extracted program logic definitions, and runs the assembled applications entirely on the local device.

Applicant's invention does not involve downloading executable code for the application - as Rice does in order in "fat client" mode. Nor does the invention download a terminal emulation thin client executable program that works with applications running on a server - as Rice does in "thin client" mode.

Applicant's invention is not directed the general ability to download executable code or text files, as that is well-known prior art that existed long ago. Showing only that both Applicant and Rice reference the general practice of "downloading some kind of file" is not a valid basis for arguing that Rice discloses the specific system and method defined in independent claims 1, 8, and 16.

The use of the "thin-client" terminology may be a possible cause of confusion. Applicant has described what has now become known as a "rich" or "smart" client as opposed to a "thin" or "fat/thick" client. However, at the time of filing of the present application, the terms "rich client" and "smart client" were not in common usage. Only the choices of "thin client" and "fat/thick client" terminology were

in common use at the time. (See, Wikipedia article at: http://en.wikipedia.org/wiki/Rich_client). In part, this says "Starting around 2003 the term 'rich client' has taken on a different meaning than thick client. It has come to mean a hybrid of 'thick' and 'thin', or an architecture where the amount of client vs. server CPU utilization is more balanced."

Applicant teaches a new and previously unknown method for creating such a rich client. The invention shares many of the same benefits as a thin client, but it differs in functionality. The pure "thin client" that Rice utilizes is basically just a screen painter (like telnet and Citrix), not a complete "smart" application that can run by itself when not connected to a remote server (which the present invention can do). Rice makes this clear where it states that the **executable code of the application itself** must be downloaded in order to run the application without a connection to a server, thus turning the local device into a true "fat client". Rice operates in either purely thin or purely fat modes. Nowhere does Rice use or teach a completely different mode (rich client or smart client) as the Applicant describes and claims.

Both thin clients and the present invention provide the distinct benefit that "fat" (large size) executable programs do not have to be pre-installed on (or downloaded to) a client device in order to enable the user to use a complete working application. Applicant used the term "thin client" at the time of filing to highlight the invention's benefits in comparison to "fat client" applications. Today, one skilled in the art would probably use the more recent terminology "smart client" or "rich client" to emphasize that the invention is different from both pure thin and pure fat clients such as Rice utilizes.

CLAIM 1

1. A system for deploying applications over a distributed network to an Internet-enabled device for interacting with a server, the server being in communication with the distributed network and having text files containing application logic, the system comprising:

an application assembler for storing on and running on the Internet-enabled device, **the application assembler for downloading one or more text files from the server, retrieving program logic from each of the downloaded text files**, and **assembling the retrieved program logic into a functioning application** and running the functioning application on the Internet-enabled device regardless of

> whether the Internet-enabled device remains connected to the server. (Emphasis added).

The Office Action states that claim 1 is anticipated by Rice, and points to the following references in Rice with some short quoted phrases, each of which will be discussed here.

Col 3: 0015 "...continues to be downloaded..."

Here is the full text of Rice paragraph 0015, with key phrases in bold:

> Enabling code is provided to enable and/or initiate the interface between a remote computing device and a local computing device. This enabling code is **downloadable in a format suitable for making possible substantially instant utilization** of the enabling code, even while additional portions of the enabling code continue to be downloaded to the local computing device. This enabling code will preferably allow **terminal emulation by a client computer**, as well as allow **further executable code to be downloaded** in the background, in order to support some level of fat-client support for a software application which may be used by the client user.

What Rice discusses is downloading of **executable** code. Rice is not unique - prior art (such as Java applets) have been doing that since at least the early 90's. Rice is merely using the downloading of executable code as a step in a system of providing terminal emulation access to remote applications together with the heart of Rice's invention, which involves links ("AppLinks") to documents and the server applications that create, modify and view those documents. This has nothing to do with Applicant's invention, which is a new way of downloading, assembling and running applications on a local device. Applicant's invention does not involve downloading executable application code as Rice does.

Nowhere does Rice discuss the downloading of **text files containing the program logic that will be assembled into a working application on the Device**. The highlighted phrases such as "**further executable code to be downloaded**" clearly show that Rice is only discussing downloading of EXECUTABLE code, like a Java applet does. Rice teaches downloading of executable code that provides

terminal emulation while allowing further downloading of other executable code that is the application itself (to permit running it in "fat client" mode on the local device instead of via terminal emulation on the server).

Also, this "further executable code" is operating system dependent, so that, under Rice, separate sets of executable application program code would be needed for each type of local device operating system to which it might be downloaded, again unlike how Applicant's invention works.

This is very different from what Applicant's invention does, which is to download text files that are operating system INDEPENDENT, retrieve abstract program logic information from those text files, assemble a working application in memory on the local device, and then run that application on the local device. Nowhere does Rice teach this in paragraph 0015. Rice only downloads executables.

Col 3: 0016 "...continues to be downloaded..."

Here is the full text of Rice paragraph 0016, with key phrases in bold:

> In a preferred embodiment of the subject invention, the enabling code provides a **terminal emulator by which the client computer may access application software resident on a server**. Preferably, **the server computer which executes the application program** transmits presentation data to the client computer on an instantaneous or near instantaneous basis, so that the terminal emulation is transparent to the client user, i.e., so that the experience of the user is similar or identical to using a fat client executable application. **As the enabling code and other software continues to be downloaded** from the remote computing device to the local computing device, a user is free to use other data such as application or file data, resident at or through either the remote computing device or the local computing device. In one embodiment of the invention, following completion of the download of enabling data of the user is, optionally, advised of a transition sequence automatically occurring to terminate the application or data download phase and fully operate within either a local computing device mode or a mode in which files are shared with the local computing device via the remote computing device.

Here again, Rice clearly describes "**application software resident on a server**", and that what is downloaded is executable program code for a terminal emulator (and perhaps also for the

application itself), not text files that contain abstract program logic that will be assembled into an application ("enabling code and other software continues to be downloaded"). All the same points made about paragraph 0015 also apply here.

Col 3: 0083 "...continues to be downloaded..."

Here is the full text of Rice paragraph 0083, with key phrases in bold.

> In one embodiment of the present invention, it is desired to provide a **common enabling code means for rapid download from communications nets, such as internets**, the worldwide web, or other providers of such communication download services. Such enabling code means is designed to provide a facility for download and upload of data from such communications nets to and from desktop, laptop, hand held, or other computing devices of various sizes and capacities. However, such enabling code means is designed with substantial commonality and interoperability with vast amounts of application and other types of software which is freely available on such communication nets.

The quoted phrase cited by the Office Action is not found in paragraph 0083, so it is not clear why paragraph 0083 was cited. There is nothing in this section that pertains to the points discussed in paragraphs 0015 and 0016 above, other than a general reference to downloading code over the internet.

Col 16: 0148 "...as a web browser plug-in..."

Here is the full text of Rice paragraph 0148, with key phrases in bold.

> FIG. 9 is a depiction of the data flows relating to the transmission of an application link according to one embodiment of the present invention. Following the creation 712 of the AppLink as described above, the AppLink network location, or URL, can be communicated as shown at 616 in FIG. 6 to a recipient 620 by, for example, and as detailed in FIG. 9, including it in an e-mail 912, i.e., an SMTP transmission, such as in an e-mail either as a hyperlink or hyperlinked graphic, or embedding it in a web page 914, e.g., providing a hyperlink or hyperlinked graphic within a web page viewable by the recipient. Other ways in which the AppLink may be transmitted to a recipient include communicating it via fax 916, a POTS telephone message or page 918 or face-to-face conversation, or

> any other means of communication 920, including surface mail, media broadcast or dedicated line. **The user interface for creation of the applink is preferably implemented as a web-based form generated by the applink server** depicted in FIG. 3 and 3b above. Returning to FIG. 8, this web-based form 812 is used to communicate with the AppLink Server 602. **Alternatively, a native program 814 capable of creating and/or receiving and viewing applinks may be resident and executed on the sender's local computer 614, as a part of an e-mail program, as a web browser plug-in 816 or a standard feature built into the browser**, or as a component of the sender's local computer operating system. More broadly, the AppLink creator can use various means to communicate with the AppLink Server and create the AppLink. The recipient's thin-client can be made accessible to the recipient in various ways. **A java applet or other download-and-run program could be downloaded just before use if the recipient does not yet have a compatible thin client available on his local computer**. A software company may want to make a thin client an intrinsic part of a web browser or operating system, as shown at 818, in order to help ensure that its thin client design enjoys wide currency. A company might want to make the thin client a part of the circuitry of the client computing device for performance purposes, also helping to ensure that it cannot be easily changed by the user to the thin client offered by another company.

Rice describes various ways to get **executable** programs onto a local device that can create, receive and view Rice's "AppLinks". The quoted phrases refer to existing prior art such as browser plugins, downloadable Java applets, and similar means of getting executable code installed on or downloaded to a local device. In fact, Rice mentions them only as ways of getting alternative (non-preferred) thin-client executable code such as Citrix or telnet programs onto a local device.

Rice's preferred means is using a web-based form generated by a remote server to create AppLinks. Applicant's invention has nothing to do with creating or using AppLinks or the server-based applications to which they refer.

Nowhere in paragraph 0148 does Rice teach the downloading of a text file containing abstract high-level program logic, the retrieval of the program logic by an AVM, the assembly of an

application on the local device by an AVM, and the running of the created application by an AVM, all of which the AVM in our invention provide. The quoted phrases in Rice all refer to **downloading executable code**.

Col 7:0096-0087(0097?) thin client executable code can be downloaded

Here is the full text of Rice paragraphs 0096-0097, with key phrases in caps.

> [0096] **in an alternate embodiment of the present invention, a thin-client is provided without a corresponding fat client mode, but data file collaboration and other related file transfer techniques are available to thin client users**. This embodiment would allow users to use alternate terminal emulation hardware, including, but not limited to, web appliances, personal digital assistants, palmtops, and any other devices which are web- or Internet-enabled, or may operate over TCP/IP, as is known in the art.
>
> [0097] **in an alternate embodiment of the subject invention, upon execution of a hyperlink request for a document by a remote user, the user may be presented with a thin client executable code which permits viewing and editing of the data file in its native format**, replacing the prior modes of data submission such as CGI forms. The data file submitting by a remote user in this way may later be viewed and edited by other remote users, the form originator, or the user submitting data through the native thin-client executable.

Again, Rice explicitly refers to "**thin client executable code**" for doing terminal emulation. All the same discussion as above applies. Nowhere in these paragraphs does Rice teach the downloading of a text file containing abstract high-level program logic, the retrieval of the program logic by an AVM, the assembly of an application on the local device by an AVM, and the running of the created application by an AVM, all of which the AVM in our invention provide.

Col 9:0107-0109 thin client executable code can be downloaded.

Here is the full text of Rice paragraphs 0107-0109, with key phrases in bold.

> [0107] FIG. 2 is a schematic illustration of a network according to an embodiment of the present invention. As depicted in FIG. 2, an e-mail or a web page can have a hyperlink to a particular application file or document stored remotely on server 140 or remote machine 144. The machines form a virtual network linked by the Internet or other communications network indicated by network cloud 148. When the recipient or user using client computer 140 clicks on or executes the link, information embedded therein directs the client computer to a server computer 140 on which a data file and user application reside, **a thin-client is automatically downloaded to the user, and the application or program associated with the file (typically the application that was used to create it, i.e., the data file's native application) is started on a remote application server computer 142.** The specific file designated by the link is then opened by the user application, and the thin-client displays the file on the user's computer 140 within a thin client window which contains the application's graphical user interface. The user is then free to interact with the file as if it were running inside an application running on the user's client PC 140. Alternatively, a remote file on a different remote computer 144 may be accessed by a user application running on server 142.
>
> [0108] According to an embodiment of the present invention, a method is provided by which a user, e.g. a remote user, may access a remote document in thin client mode, **by accessing a hyperlink to a url or network location at which an application accessing the remote file may be remotely manipulated and operated**. This hyperlink may be referred to generally herein by the generic term Application File Hyperlink, or by the trade name "AppLink.TM.".
>
> [0109] FIG. 3 is a screen view of one embodiment of a computerized method according to the present invention. A hyperlink according to the present invention may be created in a GUI environment such as that depicted in FIG. 3, showing a screen shot 150 of a GUI allowing a user to create a hyperlink to a server-supported AppLink. Alternatively, a user

> may create a hyperlink according to the present invention by executing a GUI icon representing the link-creating application, as depicted in FIGS. 3e and 3f. A client user may create a server application link by specifying a file resident on a local client machine, e.g., client machine 140 or FIG. 2, a server 142, or a remote client machine 144 which may establish from time to time a direct or indirect link to the server 140. A hyperlink to the **server-supported application** may contain an embedded URL of a server application or protocol which may execute a series of instructions to download middleware, or web-enabling client code, to the machine 140 from which the hyperlink is accessed. In addition, the hyperlink may contain embedded information, or the URL of a protocol file containing information as to the executable code to run on the server 140, as well as a particular data file to open within the **executable application being run on the server 140.** This application's output may then be sent as a "picture" by the server middleware to a web-enabling client resident on the client local machine 140, as depicted by process 118 of FIG. 1. In other words, upon execution of the hyperlink by a remote client 140, a process will be executed by the server 142 by which a remote client 140 will have a thin-client process loaded onto it, and then access the remote data file, while the native application of the data file runs on the server. The remote user interacts with the server application using middleware server and client components resident on the respective machines. **During this process, user application executable code may be downloading** as bandwidth permits, as indicted by 122 in FIG. 1."

Again, Rice explicitly says that "**a thin-client is ... downloaded**" for doing terminal emulation to a "**server-supported application**", "**(... the data file's native application) is started on a remote application server**", and "**during this process, user application executable code may be downloading**". All the same discussion as above applies. This teaches running an application on a server via thin client terminal emulation software that is downloaded as executable code (or is already resident on the local device), and maybe also downloading the user application itself, again only in an executable code format. Nowhere in these paragraphs does Rice teach the downloading of a text file containing abstract high-level program logic, the retrieval of the program logic from these text files by an AVM, the assembly

of an application on the local device by an AVM, and the running of the created application by an AVM, all of which Applicant's invention does.

CLAIM 8

8. A system for deploying an application over a network to an Internet-enabled device, the network having a server containing one or more application logic files, the application logic files containing embedded application logic relating to a computer program, the system comprising:

a program assembler for storing on and running on the Internet-enabled device, the program assembler for **downloading application logic files, retrieving embedded application logic from the application logic files**, and **building the computer program from the retrieved embedded application logic**, and **running the computer program on the Internet-enabled device**. (Emphasis added).

Again, the Office Action states that this claim is anticipated by Rice and points to the following references in Rice with some short quoted phrases, each of which will be discussed here.

Col 7:0096"...web-or Internet-enabled..." (See full text for this paragraph above)

Both Applicant and Rice deploy something to web/internet-enabled devices, but that general concept is not what Applicant claims. Rice does not teach the specific system and method of deploying applications to such a device in the manner which Applicant's invention uses, as discussed at length above.

Fig 6 and related text

Rice's FIG. 6 refers to existing commercial third-party software: "Tarantella, Citrix, etc. to open appropriate application and load file" and then deploys a thin client executable program specific to such commercial application onto the local device, if one is not already resident on the local device by some other deployment means.

Again here Rice discusses how to deploy a "thin client executable program" to a local device from a remote server. Nowhere in Fig 6 is there a representation of text files containing program logic that are resident on the server and that are downloaded to the local device for subsequent use by an

AVM to construct and run an application on the local device. Fig 6 only shows downloading executable program code.

Col 10:0111 "...have several embedded 'macros' which would be..."

Here is the partial text of Rice paragraph 0111, with key phrases in bold.

> One advantage of sending this hyperlink to a recipient in an e-mail compared to the alternative of simply including the file as an attachment to the e-mail is that there is no computer virus danger for the recipient. [...] Viewing a file regardless of the application used to create it is also the function of ADOBE's PDF document format. With PDF, a PDF file is derived from a document by a conversion utility. This file is then sent as an attachment or download to the recipient, who then views it if he has the PDF viewer installed on his computer. PDF is essentially a "picture" of a document, meaning that the recipient cannot change it. With an application file hyperlink, the recipient has full capability to interact with the file. For example, **a spreadsheet may have several embedded "macros" which would be inaccessible to a person viewing a pdf file.**

The cited phase is out of context and does not even describe any part of Rice's invention. Rice is contrasting the harmlessness of viewing a spreadsheet's data when contained within a PDF document (which is part of an existing commercial program and not an invention of Rice), as compared to the risks of downloading the actual spreadsheet and running an actual spreadsheet program that might contain dangerous macros.

Paragraph 0111 has nothing to do with downloading files that contain application program logic that is assembled and run by an AVM on the local device. Rice is not even discussing downloading files that contain macros. The quoted phrase is not relevant.

Col 3: 0015 "...continues to be downloaded..." Col 3: 0016 "...continues to be downloaded..." Col 3: 0083 "...continues to be downloaded..."

(See full text for these paragraphs above)

The same arguments as were made above for Claim 1 are fully pertinent here as well. Rice does not teach what Applicant claims in these paragraphs. Rice only discusses using the existing prior art of downloading executable code, similar to a Java applet or ActiveX object. So the references are not relevant.

Col 9: 0109 "...may contain embedded information..."

Here is the partial text of Rice paragraph 0109, with key phrases in bold.

> **A hyperlink to the server-supported application** may contain an **embedded URL of a server application** or protocol which may execute a series of instructions to download middleware, or web-enabling client code, to the machine 140 from which the hyperlink is accessed. In addition, **the hyperlink may contain embedded information, or the URL of a protocol file** containing information as to the executable code to run on the server 140, as well as a particular data file to open within the **executable application being run on the server 140**.

Paragraph 0109 says nothing about "retrieving embedded application logic from the application logic files" as claim 8 does. Rice is talking about embedding URLs of server applications or protocols (i.e. server/middleware addresses and other parameters) to request the running of server-based applications for a document. A similar example is found in many search engines (such as Google) which embed a target URL as a parameter inside a Google URL that enables Google to track "hits" on the search target sites before finally rerouting the user's web browser to the target site.

Embedding URLs in a hyperlink is very different from the embedding of program logic in a text file on a server which is then downloaded, the program logic extracted by an AVM, assembled into an application and run by an AVM.

Col 16: 0148 "...download-and-run program..."

(Full text above).

See the previous discussion of this paragraph regarding Claim 1. The same arguments are valid here.

Col 9: 0107 "... information embedded..."

Here is the partial text of Rice paragraph 0107, with key phrases in bold.

> 0107: When the recipient or user using client computer 140 **clicks on or executes the link, information embedded therein** directs the client computer to a server computer 140 on which a data file and user application reside. Here Rice is again discussing embedding a URL or similar address in a hyperlink, it has nothing to do with embedding program logic in a file on a server. Similar arguments as for 0109 above apply here as well. The cited phrase is not relevant.

CLAIM 16

16. A method for deploying a computer program over a network, the method comprising:
storing and running a software module on a client device of a user;
providing to the client device text files containing embedded application program logic for the software module, the **text files containing embedded program logic for the computer program to the installed software module upon request**;
running the computer program assembled from the embedded progam logic on the client device; and
enabling user interaction with the computer program running on the client device. (Emphasis added).

Again, the Office Action says that this claim is anticipated by Rice, and that "Rice discloses A method for deploying a computer program over a network".

However, none of the cited quoted sections from Rice describe what our invention comprises. There are many ways of deploying a computer program over a network that predate both Rice and Applicant, but not in the specific manner of Applicant's invention, which is not taught by Rice in any of the cited sections. Some words and phrases common to both patents are cited (out of context), but the invention itself is not taught by Rice.

Col 16: 0148 "...download-and-run program..." Col 16: 0148 "...as a web browser plug-in..."

(Full text above).

See the previous discussion of this paragraph in Claim 1. The same arguments are valid here.

Col 8: 0100

Here is the partial text of Rice pargarph 0100, with key phrases in bold.

> Thus, the present invention allows a remote user to have virtually instant use of an application on the server, while the **executable code** for the application itself **is being transmitted or downloaded** to the user's client machine. **Upon download of the application code, the application code may be installed** on the remote user's machine.

Here, Rice is again teaching the downloading of the executable code that comprises the application, not the downloading of text files containing abstract program logic that is assembled into a working application by an AVM.

Col 3: 0015, 0016, 0083 "...continues to be downloaded...

This is cited in regard to a key element of the claim: "providing to the client device text files containing embedded application program logic for the software module, the text files containing embedded program logic for the computer program to the installed software module upon request". The Office Action only cites this and the following paragraph 0100 from Rice, neither of which are relevant.

Here, Rice is talking about downloading EXECUTABLE CODE, not "text files containing embedded program logic". Paragraph 0100 is not relevent, because Rice does not teach downloading of such text files.

Col 7: 0097-0098 "...remote user's data file..."

The full sentence from which the quote is cited says:

> Upon execution of the instruction request 116 at the server application level, the server web-enabling application may modify the **GUI Or other interface state of the remote user's data file, indicated by 118, as**

> **returned by the "user application" running on the server**, i.e., the application that the remote user wishes to use for manipulation of a certain data file.

The "GUI State" (i.e., screen painted image of the terminal emulator) of the "remote data file" "returned by the user application running on the server" has nothing to do with "text files containing embedded program logic". This paragraph is not relevant.

Col 8: 0100-0102 ("...After installation of the software...").

Here is the partial text of Rice paragraphs 0100-0102, with key phrases in bold.

> After installation of the software, if the **software executable** is run by the user, the user will be operating in fat client mode

Again, Rice here describes running executable code that has been downloaded, not downloading text files that contain logic that is extracted, assembled and run by an AVM.

DEPENDENT CLAIMS:

Since all the rejections of the dependent claims rely on Rice, those rejections should be withdrawn for the same reasons discussed above.

Further, comments follow below for some of the dependent claims.

CLAIMS 3 AND 12

The Office Action states that Rice Col 15: 0145 discloses where the embedded application logic is operating system independent. However, it says no such thing. Rice merely says that the operating system etc. is **immaterial** if the application is being run on a server and accessed on the local device via terminal emulation. The actual operating system of each such server requires a separate code set for each such operating system. This is not OpSys independence. In addition, when Rice discusses downloading the "executable code" that comprises the application programs, this executable code by definition is operating system **dependent**. So Rice does not disclose this at all, in either the thin client or fat client modes of operation.

CLAIMS 6 AND 7

These claims are rejected "as being unpatentable over Rice in view of Lloyd." The Office Action cites Lloyd Col 9:27-67 as disclosing a script engine for interpreting scripts contained in the extracted program logic. The relevent lines are:

> Web software 404 also includes Common Gateway Interface (CGI) scripts 405 to create dynamic HTML files. **A CGI script is an external application executed by a web server. A CGI script is invoked when a user submits a request for dynamic information.**

However, Lloyd does not describe a script engine on a local device. Rather, it describes a script engine on a web server, which is only invoked when a user submits a request to the web server for dynamic information. This is not the method in which a script engine is used in the present invention.

The Office Action further states "it would have been obvious to a person of ordinary skill in the art at the time of the invention to include a parser for extracting program logic to Rice's invention." However, since Rice's invention clearly does not resemble the present invention in any important aspects, this assumption cannot be true. Rice does not teach anything like the present invention and so there is no way that someone of ordinary skill would be able to add a script engine to Rice's invention to arrive at the present invention. Rice plus a script engine does not add up to the present invention.

The Office Action further cites Lloyd Col 17: 19-67 as disclosing a component layout handler. However Lloyd does nothing of the kind. The Lloyd patent only discusses how to embed images into e-mail documents. It does not teach how to construct a general purpose layout manager for doing layout of user input components such as labels, textfields, buttons, combo boxes, tables, menubars, etc. that are specified in abstract form in a text file that is downloaded from a server, then assembled and run by an AVM. In summary, Rice and Lloyd do not teach or suggest the present invention.

Claims 1-17 and 19-22 define an invention which is neither taught nor suggested by Rice III or Lloyd. The rejection of the claims in the Office Action of February 13, 2006, should be withdrawn and all claims allowed. Notice to that effect is requested. Respectfully submited.

The Commissioner is authorized to charge any additional fees associated with this paper or credit any overpayment to Deposit Account No. 11-0982.

Respectfully submitted,

KINNEY & LANGE, P.A.

Date: 4/13/06

By: _____

David R. Fairbairn, Reg. No. 26,047
THE KINNEY & LANGE BUILDING
312 South Third Street
Minneapolis, MN 55415-1002
Telephone: (612) 339-1863
Fax: (612) 339-6580

DRF:kmm